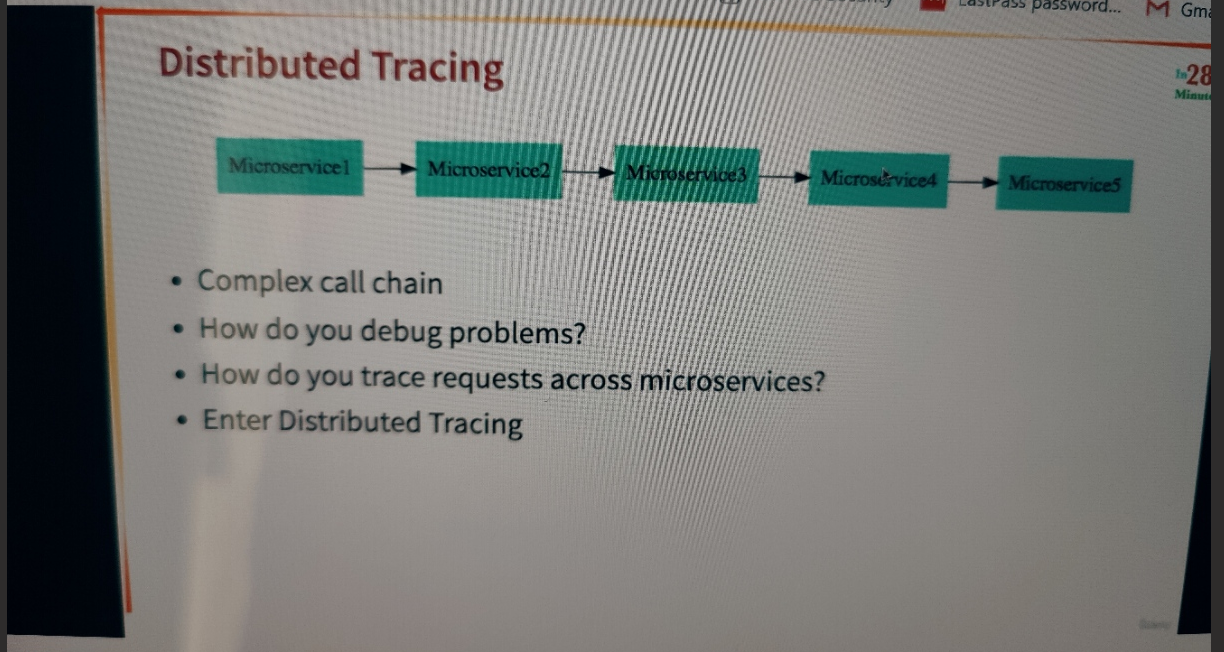
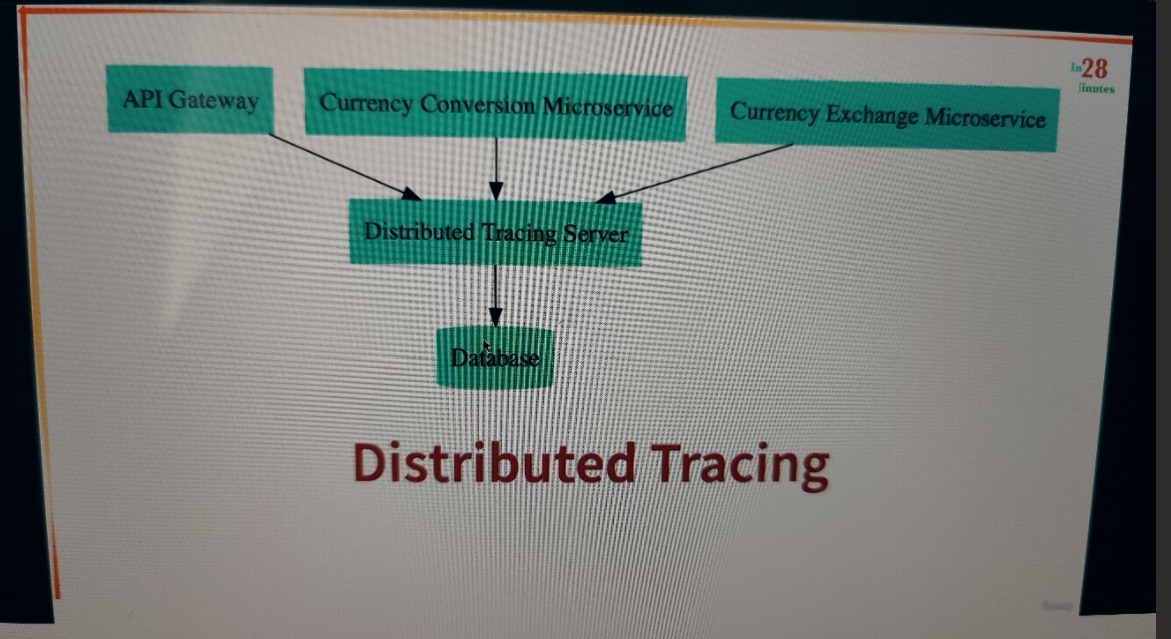
Distributed tracing:





 all the microservices involved would actually send all the information out to a single distributed tracing server, and this distributed tracing server would store everything to a database.

This database can be a in memory database or a real database.

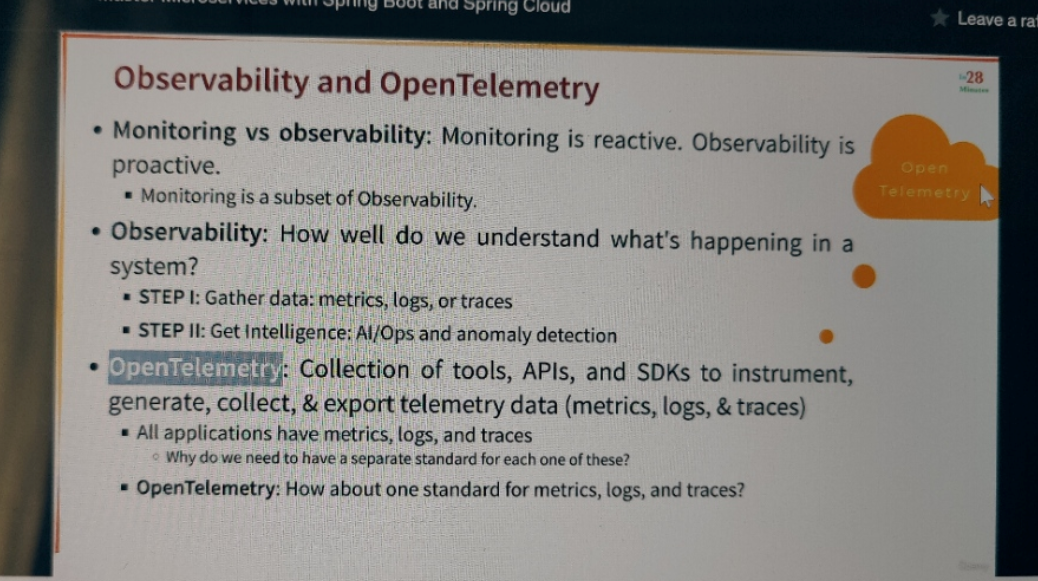
So all the information that comes out from your microservices is stored by distributed tracing server,

and the distributed tracing server would provide you an interface which would allow you to trace the request across multiple microservices.

we'll be using distributed tracing server Zipkin.

Run zipkin as a docker container

docker run -p 9411:9411 openzipkin/zipkin



Monitoring is all about looking at metrics, logs, traces. Observability, on the other hand, is proactive.

What do I mean? Observability focuses on, how well do we understand what's happening in a system? This step one in observability is gathering data: metrics, logs, or traces.

But the most important step in observability is to get the intelligence.

In monitoring, we try and get metrics, logs, and traces. And in observability, we focus on getting intelligence from this data, and hopefully this intelligence would help us to detect problems faster.

there are different standards, different tools, different APIs, different SDKs for each of metrics, logs, and traces. The question is, why do we need to have a separate standard for each one of these? One standard for metrics, one standard for logs, one standard for traces. There were a lot of initiatives around consolidating metrics, logs, and traces into a single standard. And one of the most important results from those initiatives is OpenTelemetry.

1. Add dependency

<dependency>  
 <groupId>io.micrometer</groupId>  
 <artifactId>micrometer-observation</artifactId>  
</dependency>  
  
<!-- OPTION 1: Open Telemetry as Bridge (RECOMMENDED) -->  
<!-- Open Telemetry  
 - Simplified Observability (metrics, logs, and traces) -->  
  
<dependency>  
 <groupId>io.micrometer</groupId>  
 <artifactId>micrometer-tracing-bridge-otel</artifactId>  
</dependency>  
  
<dependency>  
 <groupId>io.opentelemetry</groupId>  
 <artifactId>opentelemetry-exporter-zipkin</artifactId>  
</dependency>

Micrometer 🡪mertic.logs, traces

Open telementry 🡪 open standard for mertic, logs and tracing

1. Add prop file

# management.tracing.sampling.probability=1.0

(Sets the probability of sampling traces. A value of **1.0** means all traces are sampled. Adjust this value as needed for your tracing requirements)

# logging.pattern.level=%5p [${spring.application.name:},%X{traceId:-},%X{spanId:-}]

1. If we are using feign the tracing to microservices won't be possible

So we have to add one more dependency for that.

<!-- COMMON CHANGES + -->  
<!-- Enables tracing of REST API calls made using Feign - V3 ONLY-->  
<dependency>  
 <groupId>io.github.openfeign</groupId>  
 <artifactId>feign-micrometer</artifactId>  
</dependency>

1. Also, if directly created the restTemplate object using calling the microserivice

new RestTemplate().getForEntity()

if we are using this way tracing with zipkin wont be possible, So create the bean of restTemplate

@Configuration(proxyBeanMethods = false)  
public class RestTemplateConfiguration {  
   
 @Bean  
 RestTemplate restTemplate(RestTemplateBuilder builder)  
 {  
 return builder.build();  
 }  
}

